On the General Chain Pair Simplification Problem

Chenglin Fan[1], Omrit Filtser[*2], Matthew J. Katz[3], and Binhai Zhu[4]

1 **Montana State University**
Bozeman, MT 59717-3880, USA
bhz@montana.edu
2 **Ben-Gurion University of the Negev**
Beer-Sheva 84105, Israel
omritna@cs.bgu.ac.il
3 **Ben-Gurion University of the Negev**
Beer-Sheva 84105, Israel
matya@cs.bgu.ac.il
4 **Montana State University**
Bozeman, MT 59717-3880, USA
bhz@montana.edu

Abstract

The Chain Pair Simplification problem (CPS) was posed by Bereg et al. who were motivated by the problem of efficiently computing and visualizing the structural resemblance between a pair of protein backbones. In this problem, given two polygonal chains of lengths n and m, the goal is to simplify both of them simultaneously, so that the lengths of the resulting simplifications as well as the discrete Fréchet distance between them are bounded. When the vertices of the simplifications are arbitrary (i.e., not necessarily from the original chains), the problem is called General CPS (GCPS).

In this paper we consider for the first time the complexity of GCPS under both the discrete Fréchet distance (GCPS-3F) and the Hausdorff distance (GCPS-2H). (In the former version, the quality of the two simplifications is measured by the discrete Fréchet distance, and in the latter version it is measured by the Hausdorff distance.) We prove that GCPS-3F is polynomially solvable, by presenting an $\widetilde{O}((n+m)^6 \min\{n,m\})$ time algorithm for the corresponding minimization problem. We also present an $O((n+m)^4)$ 2-approximation algorithm for the problem. On the other hand, we show that GCPS-2H is **NP**-complete, and present an approximation algorithm for the problem.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Chain simplification, discrete Fréchet distance, dynamic programming, geometric arrangements, protein structural resemblance

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.37

1 Introduction

Polygonal curves play an important role in many applied areas, such as 3D modeling, map matching, and protein backbone structural alignment and comparison. There exist many methods for comparing curves in these (and in many other) applications, where one of the more prevalent methods is the Fréchet distance.

* Work by O. Filtser was supported by the Ministry of Science, Technology & Space, Israel, and by the Lynn and William Frankel Center.

The *Fréchet distance* between two curves is often described through the man-dog analogy. Imagine a man and a dog connected by a leash, each walking along his own curve from its starting point to its end point. Both of them can control their speed, but they can only move forward. The Fréchet distance between the two curves is the length of a minimum-length leash that allows them to reach the end point of their curves.

In the *discrete Fréchet distance* we are given finite sequences of points instead of continuous curves. The same rules apply, but now the man and the dog are hopping between the points of their sequence. The discrete Fréchet distance is a simpler version, and is considered a good approximation of the continuous distance.

Recently, the discrete Fréchet distance was used to align and compare protein backbones, yielding favorable results in many instances [11, 12]. A protein backbone may consists of as many as 500∼600 α-carbon atoms, which are the vertices (i.e., points) of our chain. Thus, a natural approach to accelerate computations is to use a simplification of the chain. In general, given a chain A of n vertices, a simplification of A is a chain A' such that A' is "close" to A and the number of vertices in A' is significantly smaller than n. The vertices of the simplification A' can be arbitrary, or restricted to the vertices of A (in order).

Simplifying two aligned chains independently does not necessarily preserve the resemblance between them. Thus, the following question arises: Is it possible to simplify both chains in a way that will retain the resemblance between them? This question has led Bereg et al. [3] to pose the Chain Pair Simplification problem (CPS). In this problem, the goal is to simplify both chains simultaneously, so that the discrete Fréchet distance between the resulting simplifications is bounded. More precisely, given two chains A and B of lengths n and m, respectively, an integer k and three real numbers δ_1,δ_2,δ_3, one needs to find two chains A',B' with vertices from A,B, respectively, each of length at most k, such that $d_1(A, A') \leq \delta_1$, $d_2(B, B') \leq \delta_2$, $d_{dF}(A', B') \leq \delta_3$ (d_1 and d_2 can be any similarity measures and d_{dF} is the discrete Fréchet distance).

When the chains are simplified using the Hausdorff distance, i.e., d_1, d_2 is the Hausdorff distance (CPS-2H), the problem becomes **NP**-complete [3]. When the chains are simplified using the Fréchet distance, i.e., d_1, d_2 is the Fréchet distance (CPS-3F), the problem is polynomially solvable, as shown by Fan et al. [9] who presented an $O(m^2n^2 \min\{m, n\})$-time algorithm for the minimization problem of CPS-3F.

In this paper we consider, for the first time, the problem where the vertices of the simplifications A', B' may be arbitrary points, Steiner points, i.e., they are not necessarily from A, B, respectively. Since this problem is more general, we call it General CPS, or GCPS for short. Our main contribution, see below, is a (relatively) efficient polynomial-time algorithm for GCPS, or more precisely, for its corresponding optimization problem. As a first step towards devising such an algorithm, we had to characterize the structure of a solution to the problem. This was quite difficult, since on the one hand, we have full freedom in determining the vertices of the simplifications, but, on the other hand, the definition of the problem induces an implicit dependency between the two simplifications. The second challenge in devising such an algorithm, is to reduce its time complexity (which is unavoidably high), by making some non-trivial observations on the combinatorial complexity of an arrangement of complex objects that arises, and by applying some sophisticated tricks.

Since the time complexity of our algorithm is still rather high, it makes sense to resort to more realistic approximation algorithms. See below for a detailed description of our results in this direction and of the rest of our results.

Related work

The Fréchet distance and its variants have been studied extensively in the past two decades. Given two polygonal curves of lengths m and n, Alt and Godau [2] gave an $O(mn \log mn)$-time algorithm for computing the Fréchet distance between them. This result in the plane was recently improved by Buchin et al [6]. The discrete Fréchet distance was originally defined by Eiter and Mannila [8], who also presented an $O(mn)$-time algorithm for computing it. A slightly sub-quadratic algorithm was given recently by Agarwal et al. [1]. Bringmann [4], and later Bringmann and Mulzer [5], presented a conditional lower bound implying that strongly subquadratic algorithms for the discrete Fréchet distance are unlikely to exist, even in the one-dimensional case and even if the solution may be approximated up to a factor of 1.399.

Bereg et al. [3] were the first to study simplification problems under the discrete Fréchet distance. They considered several versions of the problem, and presented polynomial-time exact algorithms. Driemel and Har-Peled [7] presented an algorithm for finding an approximate simplification in near linear time.

Our results

In Section 3, we show that GCPS-3F is polynomially solvable by presenting a sophisticated polynomial-time algorithm for the corresponding optimization problem. In Section 4 we give an $O(m+n)^4$-time 2-approximation algorithm for the problem. In Section 5 we consider the 1-sided version of the problem and present a simpler and more efficient algorithm for this problem. Finally, in Section 6 we investigate GCPS-2H, showing that it is **NP**-complete and presenting an approximation algorithm for the problem.

2 Preliminaries

There are several equivalent definitions for the discrete Fréchet distance. In this paper, we use the one that is based on the notion of a paired walk, following [10], [3] and [7].

Let $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_m)$ be two sequences of points in $\mathbb{R}^d$. We denote by $d(a, b)$ the distance between two points $a, b \in \mathbb{R}^d$. For $1 \leq i \leq j \leq n$, we denote by $A[i, j]$ the subchain $a_i, a_{i+1}, \ldots, a_j$ of A.

A *paired walk* along A and B is a sequence of pairs (or matchings) $W = \{(A_i, B_i)\}_{i=1}^{k}$, such that $A = A_1 \cdot A_2 \cdots A_k$ and $B = B_1 \cdot B_2 \cdots B_k$, and for any i it holds that $|A_i| = 1$ or $|B_i| = 1$ (where $|A_i|, |B_i| \geq 1$). The *cost of a paired walk* W along A and B is $d_{dF}^{W}(A, B) = \max_i \max_{(a,b) \in A_i \times B_i} d(a, b)$.

The *discrete Fréchet distance* between A and B is $d_{dF}(A, B) = \min_W d_{dF}^{W}(A, B)$. A *Fréchet walk* along A and B is a paired walk W along A and B for which $d_{dF}^{W}(A, B) = d_{dF}(A, B)$.

A δ-simplification of A w.r.t. distance d_1, is a sequence of points $A' = (a'_1, \ldots, a'_k)$, such that $k \leq n$ and $d_1(A, A') \leq \delta$. The points of A' can be arbitrary (the *general* case), or a subset of the points in A appearing in the same order as in A, i.e., $A' = (a_{i_1}, \ldots, a_{i_k})$ and $i_1 \leq \cdots \leq i_k$ (the *restricted* case).

The different versions of the chain pair simplification problem are defined as follows.

▶ **Problem 1.**
Instance: Given a pair of polygonal chains A and B of lengths n and m, respectively, an integer k, and three real numbers $\delta_1, \delta_2, \delta_3 > 0$.
Problem: Does there exist a pair of chains A',B', each of at most k vertices, such that A'

is a δ_1-simplification of A w.r.t. d_1 ($d_1(A, A') \leq \delta_1$), B' is a δ_2-simplification of B w.r.t. d_2 ($d_2(B, B') \leq \delta_2$), and $d_{dF}(A', B') \leq \delta_3$?

When the vertices of the simplifications are from A and B (restricted simplifications), the problem is called CPS, and when the vertices of the simplifications are not necessarily from A and B (arbitrary simplifications), we call the problem GCPS. For each problem, we distinguish between two versions:

1. When $d_1 = d_2 = d_H$, the problems are called CPS-2H and GCPS-2H, respectively.
2. When $d_1 = d_2 = d_{dF}$, the problems are called CPS-3F and GCPS-3F, respectively.

▶ Remark. We sometimes say that a set D of disks of radius δ covers a chain C. By this we mean that there exists a partition of C into consecutive subchains $C = C_1 \cdot C_2 \cdots C_t$, such that for each $1 \leq i \leq t$ there exists a disk in D that contains all the points of C_i.

3 GCPS under the Fréchet distance

In order to solve GCPS-3F, we consider the optimization problem: Given a pair of polygonal chains A and B of lengths n and m, respectively, and three real numbers $\delta_1, \delta_2, \delta_3 > 0$, what is the smallest number k such that there exist a pair of chains A',B', each of at most k (arbitrary) vertices, for which $d_{dF}(A, A') \leq \delta_1$, $d_{dF}(B, B') \leq \delta_2$, and $d_{dF}(A', B') \leq \delta_3$?

We begin by describing some properties that are required from an optimal solution to the problem. Then, based on these properties, we are able to refine our search for the optimal solution.

3.1 What does an optimal solution look like?

Let (A', B') be an optimal solution, that is, let A' and B' be two arbitrary simplifications of A and B respectively, such that $d_{dF}(A, A') \leq \delta_1$, $d_{dF}(B, B') \leq \delta_2$, $d_{dF}(A', B') \leq \delta_3$, and $\max\{|A'|, |B'|\}$ is minimum. Moreover, we assume that the shorter of the chains A', B' is as short as possible.

Let $W_{A'B'} = \{(A'_i, B'_i)\}_{i=1}^{t}$ be a Fréchet walk along A' and B'. Notice that, by definition, for any i it holds that $|A'_i| = 1$ or $|B'_i| = 1$.

Let $W_{AA'}$ be a Fréchet walk along A and A'. Notice that unlike in regular (one-sided) simplifications, the pairs in $W_{AA'}$ may match several points from A' to a single point from A, because A' does not depend only on A but also on B' and B. Similarly, let $W_{BB'}$ be a Fréchet walk along B and B' (see Figure 1).

With each pair $(A'_i, B'_i) \in W_{A'B'}$, we associate a pair of subchains A_i of A and B_i of B, which we call a *pair component*. Assume $A'_i = A'[p, q]$, then A_i is defined as follows:

1. If $p \neq q$, then each $a'_k \in A'[p, q]$ appears as a singleton in $W_{AA'}$ (since otherwise A' can be shortened). Let A^k be the subchain of A that is matched to a'_k, i.e., $(A^k, a'_k) \in W_{AA'}$, for $k = p, \ldots, q$. Then, we set $A_i = A^p A^{p+1} \cdots A^q$.
2. If $p = q$ and a'_p appears as a singleton in $W_{AA'}$, then we set $A_i = A^p$.
3. If $p = q$ and a'_p belongs to some subchain of A' of length at least two that is matched (in $W_{AA'}$) to a single element $a_l \in A$, we set $A_i = a_l$.

The subchains $B_1, \ldots, B_t$ are defined analogously.

We need two observations. The first one is that A_i and B_i are indeed subchains (consecutive sets of points). This is simply because the matchings of the points from A'_i and B'_i in $W_{AA'}$ and $W_{BB'}$, respectively, are sub-chains, and by definition $A_i = A^p A^{p+1} \cdots A^q$ is also a consecutive set of points. The second observation is that the subchains $A_1, \ldots, A_t$ (resp. $B_1, \ldots, B_t$) are almost-disjoint, in the sense that there can be only one point a_x that belongs

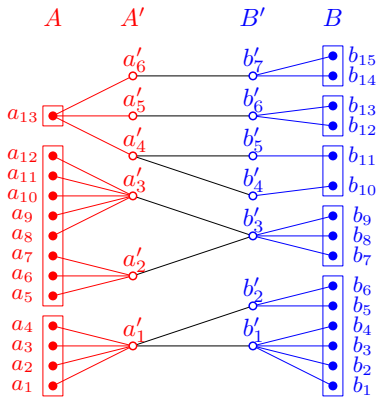


Figure 1 How does an optimal solution look like? a composition of pair-components: $W_{A'B'} = \{(\{a'_1\},\{b'_1,b'_2\}),(\{a'_2,a'_3\},\{b'_3\}),(\{a'_4\},\{b'_4,b'_5\}),(\{a'_5\},\{b'_6\}),(\{a'_6\},\{b'_7\})\}$
$(A_1 = A[1,4], B_1 = [1,6])$, $(A_2 = A[5,12], B_2 = B[7,9])$, $(A_3 = A[13], B_3 = B[10,11])$, $(A_4 = A[13], B_4 = B[12,13])$, $(A_5 = A[13], B_5 = B[14,15])$.

to both A_i and A_{i+1}, and in that case $A_i = A_{i+1} = (a_x)$. This is because if there were more than one point in common, or, if one of A_i, A_{i+1} contained more points, then the sets in $W_{AA'}$ (resp. $W_{BB'}$) were not disjoint.

So what does an optimal solution look like? It is composed of such almost-disjoint *pair-components*. A pair-component is a pair of sub-chains, (A_i, B_i), $A_i \subseteq A$, $B_i \subseteq B$, such that the points of A_i (resp. B_i) can be covered by one disk c of radius δ_1 (resp. δ_2), the points of B_i (resp. A_i) can be covered by a set C of disks of radius δ_2 (resp. δ_1), and for any $c' \in C$, the distance between the center of c and c' is at most δ_3.

The idea of the algorithm is to compute all the possible components (and that there are not too many of them), and then use dynamic programming to compute the optimal solution that is composed of pair-components.

3.2 The algorithm

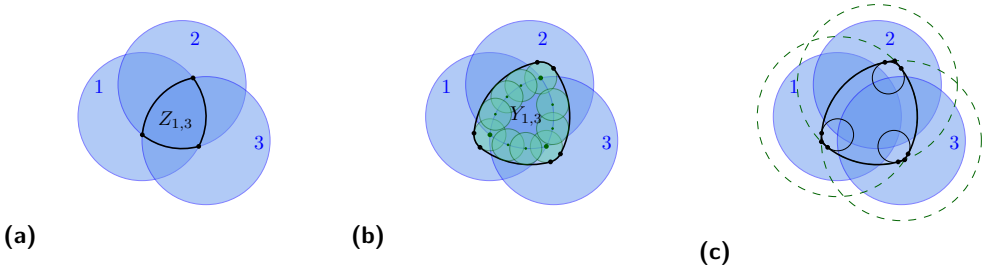
For any two sub-chains $A[i,i']$ and $B[j,j']$ there are two possible types of pair-components. In the first type, there is only one disk that covers $A[i,i']$, and in the second type, there is only one disk that covers $B[j,j']$.

We denote by $PC_1[i,i',j,j']$ the size of the minimum-cardinality set C of disks of radius δ_2 needed in order to cover $B[j,j']$, such that there exists a disk c of radius δ_1 that covers $A[i,i']$, and for any $c' \in C$, the distance between the centers of c and c' is at most δ_3. Symmetrically, we define $PC_2[i,i',j,j']$. For any 4-tuple of indices (i,i',j,j') we need to compute $PC_1[i,i',j,j']$ and $PC_2[i,i',j,j']$.

Now, in order to compute an optimal solution, we need to combine pair-components in a way that will result in a simplification of minimum size. We use dynamic programming.

Let $OPT[i,j][r]$ be the minimum number of points in a simplification of $B[1,j]$ in an optimal solution for $A[1,i], B[1,j]$ in which the number of points in the simplification of $A[1,i]$ is at most r. Then we have the following dynamic programming algorithm: $OPT[1,1][r] = 1$ if and only if $||a_1 - b_1|| \leq \delta_1 + \delta_2 + \delta_3$, and

$$OPT[1,j][r] = \min_{q \leq j}\{OPT[1,q-1][r-1] + PC_1[1,1,q,j]\},$$



(a) (b) (c)

Figure 2 The blue filled disks represent $D(b_j, \delta_2)$ and the empty dashed green disks represent $D(b_j, \delta_2 + \delta_3)$. The small disks has radius δ_3.

$$OPT[i,1][r] = \min_{p \le i}\{OPT[p-1,1][r - PC_2[p,i,1,1]] + 1\},$$

$$\begin{aligned} OPT[i,j][r] = \min_{p \le i, q \le j}\{&OPT[p-1,q-1][r-1] + PC_1[p,i,q,j], \\ &OPT[i,q-1][r-1] + PC_1[i,i,q,j], \\ &OPT[p-1,q-1][r - PC_2[p,i,q,j]] + 1, \\ &OPT[p-1,j][r - PC_2[p,i,j,j]] + 1\}. \end{aligned}$$

▶ **Theorem 1.** *For any i,j and r, $OPT[i,j][r]$ is the minimum number of points in a simplification of $B[1,j]$ in an optimal solution for $A[1,i], B[1,j]$ in which the number of points in the simplification of $A[1,i]$ is at most r.*

Proof. The proof is by induction on i, j, and r. For $i = 1$ and $j = 1$ the theorem holds by definition. Let A' and B' be an optimal solution for $A[1,i], B[1,j]$, s.t. $|A'| \le r$. Let $[p,i,q,j]$ be the last pair-component in this solution. If $[p,i,q,j]$ is of type 1, i.e. there is one disk that covers $A[p,i]$ and $PC_1[p,i,q,j]$ disks that cover $B[q,j]$, then there are two possibilities: if $p = i$ and the pair-component that came before $[p,i,q,j]$ is $[i,i,q',q-1]$ for some $q' \le q-1$, then $OPT[i,j][r] = OPT[i,q-1][r-1] + PC_1[i,i,q,j]$, else, $OPT[i,j][r] = OPT[p-1,q-1][r-1] + PC_1[p,i,q,j]$. If $[p,i,q,j]$ is of type 2, i.e. there is one point that covers $B[q,j]$ and $PC_2[p,i,q,j]$ points that cover $A[p,i]$, then again we have two possibilities, $OPT[i,j][r] = OPT[p-1,j][r - PC_2[p,i,j,j]] + 1$ or $OPT[i,j][r] = OPT[p-1,q-1][r - PC_2[p,i,q,j]] + 1$. ◀

3.3 Computing the components

Let $D(p,\delta)$ denote the disk centred at p with radius δ.

Recall that $PC_1[i,i',j,j']$ is the size of a minimum-cardinality set C of disks of radius δ_2 needed in order to cover $B[j,j']$, such that there exists a disk c of radius δ_1 that covers $A[i,i']$, and for any $c' \in C$, the distance between the centers of c and c' is at most δ_3.

We show how to find $PC_1[i,i',j,j']$ for all $1 \le i \le i' \le n$ and $1 \le j \le j' \le m$ ($PC_2[i,i',j,j']$ is symmetric). We begin with a few observations to give an intuition for the algorithm.

Consider $PC_1[i,i',j,j']$. First, notice that the center of c is in the region $X_{i,i'} = \bigcap_{i \le k \le i'} D(a_k, \delta_1)$, because the distance from c to any point in $A[i,i']$ is at most δ_1.

Any $c' \in C$ is covering a consecutive subchain of $B[j,j']$. Thus, for any $j \le t \le t' \le j'$, the center of a disk c' that covers the subsequence $B[t,t']$ (if exists) is in the region $Z_{t,t'} = \bigcap_{t \le k \le t'} D(b_k, \delta_2)$ (see Figure 2(a)). There are $O((j'-j)^2) = O(m^2)$ such regions.

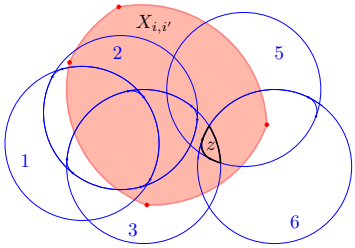


Figure 3 The arrangement $\mathcal{A}(D_A)$. After computing $Size_A(X_{1,4}, j, j')$, we know that $Size_A(X_{1,3}, j, j')$ is the minimum between $Size_A(X_{1,4}, j, j')$ and the values of the cells in $O_{1,3}$.

Each such region is convex and composed of linear number of arcs. Any point placed inside $Z_{t,t'}$ can cover $B[t, t']$, and we need a point with distance at most δ_3 to the center of c. For each $Z_{t,t'}$, consider the Minkowski sum $Y_{t,t'} = Z_{t,t'} \oplus \delta_3$ (see Figure 2(b)).

Now, consider the arrangement obtained by the intersection of $X_{i,i'}$ and the arrangement of $\{Y_{t,t'} \mid j \leq t \leq t' \leq j'\}$ (see Figure 3). Each cell in this arrangement corresponds to a set of $Y_{t,t'}$'s, each has some point with distance at most δ_3 to the same points in $X_{i,i'}$. Each cell corresponds to a possible choice of the center of c, or, in other words, a possible pair-component of type 1.

We now describe an algorithm for computing $PC_1[i, i', j, j']$ for all $1 \leq i \leq i' \leq n$ and $1 \leq j \leq j' \leq m$. The algorithm is quite complex and has several sub-procedures.

Let $X = \{X_{i,i'} = \bigcap_{i \leq k \leq i'} D(a_k, \delta_1) \mid 1 \leq i \leq i' \leq n\}$. The number of shapes in X is $O(n^2)$.

Let $Y = \{Y_{j,j'} \mid 1 \leq j \leq j' \leq m,\ Z_{j,j'} \neq \emptyset\}$. The number of shapes in Y is $O(m^2)$, each shape is of complexity $O(m)$.

Consider the arrangement $\mathcal{A}(Y)$ of the shapes in Y.

▶ **Lemma 2.** *The number of cells in $\mathcal{A}(Y)$ is $O(m^4)$.*

Proof. Let P be the set of intersection points between the disks in $\{D(b_j, \delta_2) \mid 1 \leq j \leq m\}$. Consider the following set of disks: $D = \{D(b_i, \delta_2 + \delta_3) \mid 1 \leq i \leq m\} \cup \{D(p, \delta_3) \mid p \in P\}$. Notice that the arcs and vertices of $\mathcal{A}(Y)$ are a subset of the arcs and vertices of $\mathcal{A}(D)$ (see Figure 2(c)). Since the number of points in P is $O(m^2)$, we get that the number of disks in $\mathcal{A}(D)$ is $O(m^2)$, and thus the complexity of $\mathcal{A}(D)$ is $O(m^4)$. ◀

Notice that for any shape $Y_{j,j'} \in Y$ and a cell $z \in \mathcal{A}(Y)$ it holds that $Y_{j,j'} \cap z \neq \emptyset$ if and only if $z \subseteq Y_{j,j'}$. For each cell $z \in \mathcal{A}(Y)$, let Y_z be the set of $O(m^2)$ shapes from Y that contain z. The algorithm has two main steps:

1. For each cell $z \in \mathcal{A}(Y)$, and for any two indices $1 \leq j \leq j' \leq m$, compute $Size_B(z, j, j')$ – the minimum number of shapes from Y_z needed in order to cover the points of $B[j, j']$. Recall that a shape $Y_{t,t'} \in Y_z$ covers the subsequence $B[t, t']$, in other words, there exists a point q in $Y_{t,t'}$ s.t. $d(q, b_k) \leq \delta_2$ for any $t \leq k \leq t'$.
2. For each shape $X_{i,i'} \in X$, and for any two indices $1 \leq j \leq j' \leq m$, compute $Size_A(X_{i,i'}, j, j') = \min_{z \cap X_{i,i'} \neq \emptyset} Size_B(z, j, j')$.

Note that $Size_A(X_{i,i'}, j, j') = PC_1[i, i', j, j']$.

Algorithm 1 $Size_B(Y_z)$

For j from 1 to m:
1. Set $counter \leftarrow 1$
2. Set $j' \leftarrow j$.
3. Set $p \leftarrow \max\{next(Y_{j,j'}), max(j'+1)\}$.
4. While $p \neq -\infty$:
   For k from j' to p: Set $Size_B(z,j,k) \leftarrow counter$.
   Set $counter \leftarrow counter + 1$
   Set $p \leftarrow \max\{next(Y_{j',k}), max(k+1)\}$.
   Set $j' \leftarrow k$.

Step 1

First we have to find the set Y_z for each cell $z \in \mathcal{A}(Y)$. We start by computing Y: for any j, j' we check whether $\bigcap_{j \leq k \leq j'} D(b_k, \delta_2) \neq \emptyset$. If yes, we add $Y_{j,j'}$ to Y. This can be done in $O(m^3)$ time. Then we compute the arrangement $\mathcal{A}(Y)$, while maintaining the lists Y_z for any cell $z \in \mathcal{A}(Y)$. This can be done in $O(m^4)$ as the complexity of $\mathcal{A}(Y)$ is $O(m^4)$.

Now, for each cell $z \in \mathcal{A}(Y)$ we compute $Size_B(z,j,j')$ for all $1 \leq j \leq j' \leq m$ as follows: Notice that the problem of finding a minimum cover to $B[j,j']$ from a set of subsequences, is actually an interval-cover problem. We refer to the shapes in Y_z as intervals (between 1 and m), and the goal is to find the minimum number of intervals from Y_z needed in order to cover the interval $[j,j']$.

First, for every $1 \leq j \leq n$ we find $max(j)$ - the largest interval from Y_z that starts at j. This can be done simply in $O(m^2 \log m)$ time, by sorting the intervals first by their lower bound and then by their upper bound.

Next, for an interval $Y_{t,t'} \in Y_z$, consider the intervals in Y_z whose lower bound lies in $[t,t']$ and whose upper bound is greater than t'. Let $next(Y_{t,t'})$ be the largest upper bound among the upper bounds of these intervals. We can find $next(Y_{t,t'})$, for each $Y_{t,t'} \in Y_z$, in total time $O(m^2 \log m)$, using a segment tree as follows: Let $S = \{s_1, \ldots, s_n\}$ be a set of line segments on the x-axis, $s_i = [a_i, b_i]$. Construct a segment tree for the set S. With each vertex v of the tree, store a variable r_v, whose initial value is $-\infty$. Query the tree with each of the left endpoints. When querying with a_i, in each visited vertex v with non-empty list of segments do: if $b_i > r_v$, then set r_v to b_i. Finally, for each segment s, let $next(s)$ be the maximum over the values r_v of the vertices storing s.

After computing $next(Y_{t,t'})$ for all $Y_{t,t'} \in Y_z$ (assume $next(Y_{t,t'}) = -\infty$ for $Y_{t,t'} \notin Y_z$), we use Algorithm 1 to compute $Size_B(z,j,j')$ for all $1 \leq j \leq j' \leq m$. The running time of Algorithm 1 is $O(m^2)$. Thus, computing $Size_B(z,j,j')$ for all cells $z \in \mathcal{A}(Y)$ and all indices $1 \leq j \leq j' \leq m$ takes $O(m^6 \log m)$ time.

Step 2

Recall that $\mathcal{A}(Y)$ is the arrangement obtained from the shapes in Y. Let $\mathcal{A}(D_A)$ be the arrangement of the disks $D_A = \{D(a_k, \delta_1) \mid 1 \leq k \leq n\}$. The number of cells in $\mathcal{A}(D_A)$ is $O(n^2)$.

A trivial algorithm to compute the value $Size_A(X_{i,i'}, j, j')$ is by considering the values $Size_B(z,j,j')$ of $O(m^4)$ cells from $\mathcal{A}(Y)$. Since there are $O(n^2)$ shapes $X_{i,i'} \in X$ and $O(m^2)$ pairs of indices $1 \leq j \leq j' \leq m$, the running time will be $O(n^2 m^6)$. We manage to reduce

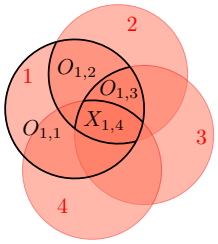


Figure 4 The arrangement $\mathcal{A}(D_A)$. After computing $Size_A(X_{1,4}, j, j')$, we know that $Size_A(X_{1,3}, j, j')$ is the minimum between $Size_A(X_{1,4}, j, j')$ and the values of the cells in $O_{1,3}$.

the running time by a factor of $O(n)$, using some properties of the arrangement of disks.

Let $\mathcal{U}$ be the arrangement of the shapes in Y and the disks in D_A. Notice that $\mathcal{U}$ is a union of the arrangements $\mathcal{A}(D_A)$ and $\mathcal{A}(Y)$.

▶ **Lemma 3.** *The number of cells in $\mathcal{U}$ is $O((m^2+n)^2)$.*

The proof is similar to the proof of Lemma 2.

We make a few quick observations:

▶ **Observation 1.** For any two cells $w \in \mathcal{U}, x \in \mathcal{A}(D_A)$, $x \cap w \neq \emptyset$ if and only if $w \subseteq x$. Similarly, for any cell $z \in \mathcal{A}(Y)$, $z \cap w \neq \emptyset$ if and only if $w \subseteq z$.

▶ **Observation 2.** For any cell $x \in \mathcal{A}(D_A)$, if $X_{i,i'} \cap x \neq \emptyset$, then $x \subseteq X_{i,i'}$.

▶ **Observation 3.** For any $1 \leq i \leq i' \leq n$ we have $X_{i,i'+1} \subseteq X_{i,i'}$.

Given $w \in \mathcal{U}$, let z_w be the cell from $\mathcal{A}(Y)$ that contains w. We have $Size_B(w, j, j') = Size_B(z, j, j')$.

Let $O_{i,i'}$ be the set of cells $w \in \mathcal{U}$ s.t. $w \subseteq X_{i,i'}$ and $w \nsubseteq X_{i,i'+1}$.

For fixed $1 \leq j \leq j' \leq m$ and $1 \leq i \leq n$, the idea is to compute the values $Size_A(X_{i,n}, j, j'), Size_A(X_{i,n-1}, j, j'), \ldots, Size_A(X_{i,i}, j, j')$ in this order, so we can use the value of $Size_A(X_{i,i'+1}, j, j')$ in order to compute $Size_A(X_{i,i'}, j, j')$, adding only the values of the cells in $O_{i,i'}$ (see Figure 4). This way, any cell in $\mathcal{U}$ will be checked only once (for any fixed $1 \leq j \leq j' \leq m$ and $1 \leq i \leq n$), and the running time will be $O(m^2 n(n+m^2)^2)$.

Now we have to show how to find the sets $O_{i,i'}$.

First, for any cell $x \in \mathcal{A}(D_A)$ we find all the cells $w \in \mathcal{U}$ such that $w \subseteq x$. There are $O(n^2)$ cells in $\mathcal{A}(D_A)$, but from Observation 1 we keep a total of $O((m^2+n)^2)$ cells from $\mathcal{U}$.

Then, for any shape $X_{i,i'} \in X$ we find the set of cells $P_{i,i'}\{x \in \mathcal{A}(D_A) \mid x \subseteq X_{i,i'}\}$. There are $O(n^2)$ shapes in X, and for each shape we keep $O(n^2)$ cells from $\mathcal{A}(D_A)$.

Now we have $O_{i,i'} = P_{i,i'} \setminus P_{i,i'+1}$. The size of $P_{i,i'}$ is $O(n^2)$, so computing $O_{i,i'}$ for all $1 \leq i \leq i' \leq n$ takes $O(n^4)$ time.

The total running time for all $PC_1[i, i', j, j']$ is $O(m^6 \log m + m^2 n(n+m^2)^2)$

Total running time

For computing $PC_2[i, i', j, j']$ we get symmetrically a total running time of $O(n^6 \log n + n^2 m(m+n^2)^2)$, so the running time for computing all the components is $\widetilde{O}((m+n)^6 \min\{m, n\})$. Calculating $OPT[i, j][r]$ takes $O(m^2 n^2 \min\{m, n\})$ time, all together takes $\widetilde{O}((m+n)^6 \min\{m, n\})$ time.

Algorithm 2

Find $X_{i,i'} = \bigcap_{i \le k \le i'} D(a_k, \delta_1)$.
Set $R \leftarrow \mathbb{R}$.
Set $counter \leftarrow 1$.
Set $k \leftarrow j$.
While $k \le j'$ and $counter \neq \infty$:
1. Set $R \leftarrow R \cap D(b_k, \delta_2)$.
2. If $(X_{i,i'} \oplus \delta_3) \cap R \neq \emptyset$, set $APC_1[i, i', j, k] \leftarrow counter$.
3. Else,
   Set $R \leftarrow D(b_k, \delta_2)$.
   If $(X_{i,i'} \oplus \delta_3) \cap R \neq \emptyset$, set $counter \leftarrow counter + 1$.
   Else, set $counter \leftarrow \infty$.
   Set $APC_1[i, i', j, k] \leftarrow counter$.
4. Set $k \leftarrow k + 1$.

4 Approximating GCPS

All the missing proofs of this section can be found in the full version of the paper.

To approximate GCPS, we use approximated pair-components which are easier to compute.

Let $APC_1[i, i', j, j']$ be the minimum number of disks with radius δ_2 needed in order to cover the points of $B[j, j']$ (in order), and whose centers are located in $X_{i,i'} \oplus \delta_3$. Similarly, let $APC_2[i, i', j, j']$ be the minimum number of disks with radius δ_1 needed in order to cover the points of $A[i, i']$ (in order), and whose centers are located in $Z_{j,j'} \oplus \delta_3$.

▶ **Lemma 4.** *For any* $1 \le i \le i' \le n, 1 \le j \le j' \le m$, $APC_1[i, i', j, j'] \le PC_1[i, i', j, j']$.

4.1 Computing the approximated components

We present a greedy algorithm that given $1 \le i \le i' \le n, 1 \le j \le j' \le m$, computes $APC_1[i, i', j, k]$ for all $j \le k \le j'$ (resp. $APC_2[i, k, j, j']$ for all $i \le k \le i'$). The algorithm runs in $O((j' - j)(j' - j + i' - i))$ time (See Algorithm 2).

Running time

Finding $X_{i,i'}$ takes $O(i' - i)$ time, and step 1 takes $O(j' - j)$ time. Step 2 takes $O(j' - j + i' - i)$ time, since the complexity of $X_{i,i'} \oplus \delta_3$ is $O(i' - i)$, the complexity of R is $O(j' - j)$, and both regions are convex. The while loop runs $O(j' - j)$ times, so the total running time is $O((j' - j)(j' - j + i' - i))$.

Computing all the approximated pair components using Algorithm 2 takes $O(n^2 m^2 (m+n))$ time. The idea of our algorithm is to compute only a small part of the components, and then approximate the others using the ones that were computed.

▶ **Lemma 5.** *Fix* $1 \le i \le i' \le n, 1 \le j \le j' \le m$, *then for any* $i \le x \le i'$ *and* $j \le y \le j'$*:*
1. $APC_1[i, x, j, j'] \le APC_1[i, i', j, j']$ *and* $APC_1[x, i', j, j'] \le APC_1[i, i', j, j']$.
2. $APC_1[i, i', j, y] + APC_1[i, i', y, j'] \le APC_1[i, i', j, j'] + 1$.
3. $APC_1[i, x, j, y] + APC_1[x, i', y, j'] \le APC_1[i, i', j, j'] + 1$.

We only compute $APC_1[i, i, j, j']$,$APC_2[i, i, j, j']$ for all $1 \le i \le n$ and $1 \le j \le j' \le m$, and $APC_1[i, i', j, j]$,$APC_2[i, i', j, j]$ for all $1 \le i \le i' \le n$ and $1 \le j \le m$. This takes $O(nm^3 + n^2 m^2)$ time using Algorithm 2.

4.2 Composing the approximated solution

Let $AAPC_1[i,i',j,j'] = APC_1[i,i,j,j'] + APC_1[i,i',j',j']$. By Lemma 5(3), choosing $x = i$ and $y = j'$, we have $APC_1[i,i,j,j'] + APC_1[i,i',j',j'] \leq APC_1[i,i',j,j'] + 1$, and by Lemma 4 we have $AAPC_1[i,i',j,j'] \leq PC_1[i,i',j,j'] + 1$.

Now let $APX[i,j]$ be the approximate solution for $A[1,i]$ and $B[1,j]$. We set

$$APX[i,j] = \min_{p<i,q<j} APX[p,q] + \min\{AAPC_1[p+1,i,q+1,j], AAPC_2[p+1,i,q+1,j]\}$$

Obviously, given the values of $AAPC_1$ and $AAPC_2$, $APX[n,m]$ can be computed in $O(m^2n^2)$ time.

▶ **Lemma 6.** *Let OPT be the size of an optimal solution, i.e. OPT is the smallest number such that there exists a pair of chains A',B' each of at most OPT (arbitrary) vertices, such that $d_1(A,A') \leq \delta_1$, $d_2(B,B') \leq \delta_2$, and $d_{dF}(A',B') \leq \delta_3$. Then $APX[n,m] \leq 2 \cdot OPT$.*

Thus we have the following theorem:

▶ **Theorem 7.** *A 2-approximation for GCPS can be computed in $O(nm^3 + n^2m^2 + n^3m)$ time.*

▶ Remark. Notice that we do not have to actually compute a solution to GCPS, just to return the minimum k. A solution of size $2 \cdot OPT$ can be computed as follows: for each approximated component $APC_1[i,i',j,j']$ (or $APC_2[i,i',j,j']$) keep the set C_1 of centers of disks that are located in $X_{i,i'} \oplus \delta_3$. For each such center $c_1 \in C_1$, find a point c_2 in $X_{i,i'}$ s.t. $d(c_1,c_2) \leq \delta_3$, and put c_2 in a new set C_2. If our solution $APX[n,m]$ uses the approximated component $APC_1[i,i',j,j']$, then the points of C_1 will be used to cover $B[j,j']$ and the points of C_2 will be used to cover $A[i,i']$.

5 1-Sided GCPS

As in [9], we consider the 1-sided variant of GCPS. In this variant we can imagine there are two dogs, one is walking on a path A and the other on a path B, and a man has to walk both of them, one with a leash of length δ_1 and the other with a leash of length δ_2. We have to find a minimum-size polygonal path for the man, such that he can walk both dogs together.

▶ **Problem 2** (1-Sided General Chain Pair Simplification).
Instance: Given a pair of polygonal chains A and B of lengths n and m, respectively, an integer k, and two real numbers $\delta_1, \delta_2 > 0$.
Problem: Does there exist a chain C of at most k (arbitrary) vertices, such that $d_{dF}(A,C) \leq \delta_1$ and $d_{dF}(B,C) \leq \delta_2$?

Denote $X_{i,i'} = \bigcap_{i \leq k \leq i'} D(a_k, \delta_1)$ and $Z_{j,j'} = \bigcap_{j \leq k \leq j'} D(b_k, \delta_2)$ as before.

For any $1 \leq i \leq i' \leq n$ and $1 \leq j \leq j' \leq m$, let $I[i,i',j,j'] = \begin{cases} 1, & X_{i,i'} \cap Z_{j,j'} \neq \emptyset \\ 0, & otherwise \end{cases}$.

Notice that $I[i,i',j,j'] = 1$ if and only if there exists one point that covers both $A[i,i']$ and $B[j,j']$. The values of $I[i,i',j,j']$ can be computed in $O((n+m)^4)$ time (the details can be found in the full version of the paper).

Now we use a dynamic programming algorithm as follows: Let $OPT[i,j]$ be the length of the minimum-length sequence C such that $d_{dF}(A[1,i],C) \leq \delta_1$ and $d_{dF}(B[1,j],C) \leq \delta_2$. Fix $i,j > 1$, we have $OPT[i,j] = \min_{p,q:I[p,i,q,j]=1} \{OPT[p-1,q-1]+1\}$.

Running time

The values of $I[i, i', j, j']$ can be computed in $O((n+m)^4)$ time. For each $i, j > 1$, we have $O(mn)$ values to check. Thus, the running time is $O((m+n)^4)$.

6 GCPS under the Hausdorff distance

The Hausdorff distance between two sets of points A and B is defined as follows:

$$d_H(A, B) = \max\{\max_{a \in A} \min_{b \in B} d(a, b),\ \max_{b \in B} \min_{a \in A} d(a, b)\}.$$

As mentioned above, the chain pair simplification under the Hausdorff distance (CPS-2H) is **NP**-complete. In this section we investigate the general version of this problem. We prove that it is also **NP**-complete, and give an approximation algorithm for the problem.

6.1 GCPS-2H is NP-complete

We show that GCPS under Hausdorff distance (GCPS-2H) is **NP**-complete, we use a simple reduction from geometric set cover: Given a set P of n points, and a radius δ, find the minimum number of disks with radius δ that cover P.

Let the sequence A be the points of P in some order (the order does not matter), and the sequence B be one point b with distance 2δ from P. Let $\delta_1 = \delta_2 = \delta$ and $\delta_3 = 4\delta + diam(P)$. Now a simplification for B is just one point anywhere in $D(b, \delta)$, and finding a simplification for A is equivalent to finding the minimum-cardinality set of disks that covers P.

▶ **Theorem 8.** *GCPS-2H is **NP**-complete.*

6.2 An approximation algorithm for GCPS-2H

Consider the variant of GCPS-2H where $d_1 = d_2 = d_H$ and the distance between the simplifications A' and B' is measured with Hausdorff distance and not Fréchet distance (i.e. $d_H(A', B') \leq \delta_3$ instead of $d_{dF}(A', B') \leq \delta_3$). We call this variant GCPS-3H, and show that GCPS-3H=GCPS-2H.

▶ **Lemma 9.** *Given two sets of points A and B, if $d_H(A, B) \leq \delta$, then there exist an ordering A' of the points in A and an ordering B' of the points in B, such that $d_{dF}(A', B') \leq \delta$.*

Proof. We construct a bipartite graph $G(V = A \cup B, E)$, where $E = \{(a, b) \mid a \in A,\ b \in B,\ d(a, b) \leq \delta\}$. Notice that since $d_H(A, B) \leq \delta$, there are no isolated vertices. Now, while there exists a path with three edges in the graph, delete the middle edge. The maximal path in the resulting graph G' has at most two edges, and there are still no isolated vertices (because we only delete the middle edge). Let $C_1, \ldots, C_t$ be the connected components of G'. Notice that each C_i has exactly one point from A or exactly one point from B. Let A' be the sequence of points $C_1 \cap A, \ldots, C_t \cap A$, and B' be the sequence $C_1 \cap B, \ldots, C_t \cap B$. We get that $C_1, \ldots, C_t$ are a paired walk along A' and B' with cost at most δ. ◀

Since we can choose the order of points in the simplifications A' and B' in the GCPS-2H problem, we get by the above lemma that any solution for GCPS-3H is also a solution for GCPS-2H. Now, since for any two sequence P, Q we have $d_H(P, Q) \leq d_{dF}(P, Q)$, we get that any solution for GCPS-2H is also a solution for GCPS-3H.

Let $S_1 = \{p_1, \ldots, p_k\}$ be the smallest set of points such that for each $a_i \in A$ there exists some $p_j \in S_1$ s.t. $d(a_i, p_j) \leq \delta_1$ and for each $p_j \in S_1$ there exists some $b_i \in B$ s.t.

$d(p_j, b_i) \leq \delta_2 + \delta_3$. Notice that since S_1 is minimum, we also know that for each $p_j \in S_1$ there exists some $a_i \in A$ s.t. $d(a_i, p_j) \leq \delta_1$ (or, we can just delete the points of S_1 that do not cover any points from A).

We can find a c-approximation for S_1, using a c-approximation algorithm for discrete unit disk cover (DUDC). The DUDC problem is defined as follows: Given a set P of t points and a set D of k unit disks on a 2-dimensional plane, find a minimum-cardinality subset $D' \subseteq D$ such that the unit disks in D' cover all the points in P. We denote by $T_c(k, t)$ the running time for a c-approximation algorithm for the DUDC problem with k unit disks and t points.

▶ **Lemma 10.** *Given a c-approximation algorithm for the DUDC problem that runs in $T_c(k, t)$ time, we can find a c-approximation for S_1 in $T_c(n, (m+n)^2) + O((m+n)^2)$ time.*

Proof. Compute the arrangement of $\{D(a_i, \delta_1)\}_{1 \leq i \leq m} \cup \{D(b_j, \delta_2 + \delta_3)\}_{1 \leq j \leq n}$ (there are $(m+n)^2$ disjoint cells in the arrangement). Clearly, it is enough to choose one candidate from each cell. Now we can use the c-approximation algorithm for the DUDC problem. ◀

Symmetrically, let $S_2 = \{q_1, \ldots, q_l\}$ be the smallest set of points such that for each $b_i \in B$ there exists some $q_j \in S_2$ s.t. $d(b_i, q_j) \leq \delta_2$ and for each $q_j \in S_2$ there exists some $a_i \in A$ s.t. $d(q_j, a_i) \leq \delta_1 + \delta_3$.

For each point $p_j \in S_1$ there exists some $b_i \in B$ s.t. $d(p_j, b_i) \leq \delta_2 + \delta_3$, so we can find a point p'_j such that $d(p'_j, b_i) \leq \delta_2$ and $d(p'_j, p_j) \leq \delta_3$. Denote $S'_1 = \{p'_1, \ldots, p'_k\}$. We do the same for the points of S_2, and find a set $S'_2 = \{q'_1, \ldots, q'_k\}$ such that for any $q'_j \in S'_2$,$d(q'_j, q_j) \leq \delta_3$ and there exists some $a_i \in A$ s.t. $d(q'_j, a_i) \leq \delta_1$.

Now, we know that for each $a_i \in A$ there exists some $p \in S_1 \cup S'_2$ s.t. $d(a_i, p) \leq \delta_1$, and, on the other hand, for each $p \in S_1 \cup S'_2$ there exists some $a_i \in A$ s.t. $d(a_i, p) \leq \delta_1$. So we have $d_H(A, S_1 \cup S'_2) \leq \delta_1$. Similarly, we have $d_H(B, S_2 \cup S'_1) \leq \delta_2$. We also know that for each $p_j \in S_1$ we have a point $p'_j \in S'_1$ s.t. $d(p'_j, p_j) \leq \delta_3$, and for each $q'_j \in S'_2$ we have a point $q_j \in S_2$ s.t. $d(q'_j, q_j) \leq \delta_3$. So we also have $d_H(S_1 \cup S'_2, S_2 \cup S'_1) \leq \delta_3$, and since CPS-2H=CPS-3H, we get that $S_1 \cup S'_2$ and $S_2 \cup S'_1$ is a possible solution for CPS-2H.

The size of the optimal solution OPT is at least $\max\{|S_1|, |S_2|\}$. Using a c-approximation algorithm for finding S_1 and S_2, the size of the approximate solution will be $c(|S_1| + |S_2|) \leq 2c \max\{|S_1| + |S_2|\} = 2c \cdot OPT$.

▶ **Theorem 11.** *Given a c-approximation algorithm for the DUDC problem that runs in $T_c(k, t)$ time, our algorithm gives a $2c$-approximation for the GCPS-2H problem, and runs in $T_c(n, (m+n)^2) + T_c(m, (m+n)^2) + O((m+n)^2)$ time.*

Acknowledgements. The authors would like to Michael Kerber for suggesting the problem.

References

1. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2):429–449, 2014.
2. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geometry Appl.*, 5:75–91, 1995.
3. Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. Simplifying 3D polygonal chains under the discrete Fréchet distance. In *Proc. 8th Latin American Theoretical Informatics Sympos., LATIN'08*, pages 630–641, 2008.
4. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th IEEE Annual Sympos. on Foundations of Computer Science, FOCS'14*, pages 661–670, 2014.

5 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *JoCG*, 7(2):46–76, 2016.

6 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog — with an application to Alt's conjecture. In *Proc. 25th Annual ACM-SIAM Sympos. on Discrete Algorithms, SODA'14*, pages 1399–1413, 2014.

7 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the Fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5):1830–1866, 2013.

8 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Information Systems Dept., Technical University of Vienna, 1994.

9 Chenglin Fan, Omrit Filtser, Matthew J. Katz, Tim Wylie, and Binhai Zhu. On the chain pair simplification problem. In *Algorithms and Data Structures - 14th Internat. Symp., WADS 2015*, pages 351–362, 2015.

10 Michael Godau. A natural metric for curves – computing the distance for polygonal chains and approximation algorithms. In *STACS 91, 8th Annual Sympos. on Theoretical Aspects of Computer Science*, pages 127–136, 1991.

11 Minghui Jiang, Ying Xu, and Binhai Zhu. Protein structure-structure alignment with discrete Fréchet distance. *J. Bioinformatics and Computational Biology*, 6(1):51–64, 2008.

12 Tim Wylie, Jun Luo, and Binhai Zhu. A practical solution for aligning and simplifying pairs of protein backbones under the discrete Fréchet distance. In *Proc. Internat. Conf. Computational Science and Its Applications, ICCSA'11, Part III*, pages 74–83, 2011.